Jets status update, May 2006

Matthew J. Sottile
matt@lanl.gov

Data Driven Modeling and Analysis Team
Los Alamos National Laboratory

May 2006

Overview of current work

- Some final work on shape metrics ala March workshop.
- Whole image metrics: geomeasures on the image
- Warping: an update
- Formation of “nuance” working group

Shape metrics revisited

The application of shape-based geomeasures was the main result from the March workshop, and it yielded a few conclusions with mixed success.

- Whole image shape metrics were indistinguishable from L2 norm due to field of view closure dominating metric.
- Explicit selection of a region of interest yielded reasonable registration, but with high sensitivity to length scale.
- *k*-means segmentation proved to be too sensitive to noise and failed to account for spatial correlations in data.
- A few red herrings were identified such as boundary parameterization.
- Sequence similarity sensitivity to starting position of boundary parameterization.

Some of these were addressed post-report.

A better length scale and ROI

The work at the March workshop given in our March PPT report had some warts. To summarize:

- A poor length scale choice: inexperience on the part of some of the participants led to geometric measures being computed at length scales near the pixel size. The noise in the data was falsely attributed to both bridging in the boundary, and sensitivity of the segmenter. The ultimate cause of problems was the radius being on the order of the discretization size.
- ROI selection. Contrary to what we had instucted participants to do, we found afterwards that the ROI included the field of view closure. This dominated the metrics.

These issues were addressed and the following data was produced.

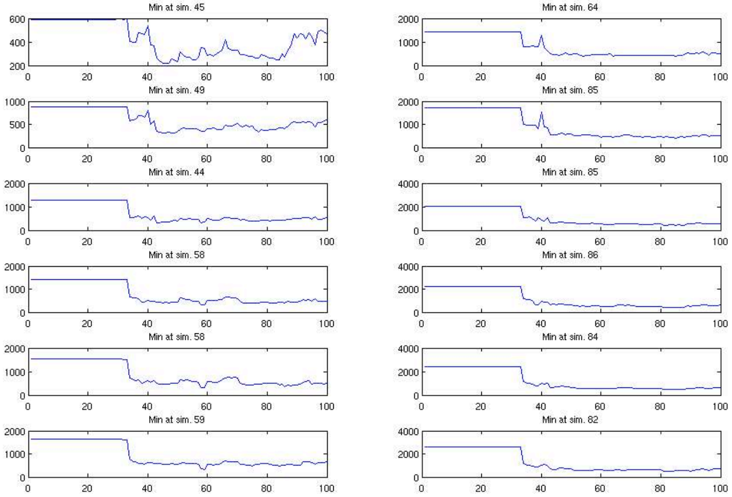
A reminder: the analysis pipeline

```
GM(TP(PADDER(EXPCROP(K3(SMASK(K4(EXPIN),EXPIN))))))
GM(TP(PADDER(SIMCROP(K3(SMASK(K4(SIMIN),SIMIN))))))

K4=kmeans(4,20)
K3=kmeans(3,10)
TP=trigpoly(pixel_width,pixel_height,df,lowpasscutoff)
GM=geomeasures2(lengthscales)
SIMCROP=crop([85 137 50 208])
EXPCROP=crop([155 235 115 275])
PADDER=pad([2 2 2 2],1)
SMASK=segmask(2);

pixel_width=1 ; pixel_height=1 ; median_window=[5 8]
lowpasscutoff=0.15 ; lengthscales=[10 13 16 19 22 25]
smallscale=[5] ; mdrtype=1 ; df=[]
```

Results with refined ROI and a sane length scale



Discussion

Better match. Best gets 6ns at sim 59, 8ns at sim 82. Within +/- 200 ps experiment time error bars. Sims vs 6ns on left, sims vs 8ns on right. Note the large dynamic range - significantly better than the DTW results reported in March with the bad length scale.

Length scale varied for each plot pair. Scales were 10, 13, 16, 19, 22, 25 pixels.

This work was run with the original simulation set. It has not been re-run on the new data yet. Unfortunately, I didn't make a plot to zoom in and show that there is less noise than bad radius DTW. Signal to noise is much better, and I can show it as soon as IDA starts to behave again.

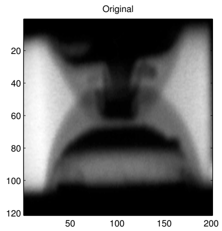
Better segmenters

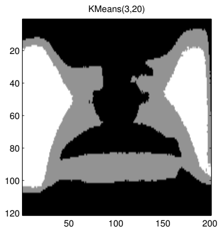
k-means proved to be too sensitive to both noise and ambiguity along the shape boundary. It didn't take into account spatial correlations between pixels.
Two replacement segmenters were implemented.

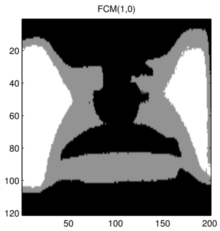
1. Fuzzy c-means: relax the constraint that pixels must be a member in one and only one segment during iteration. Pixels are allowed to have degrees of membership that are relaxed as iterations proceed.
2. Spatial fuzzy c-means: extend FCM to take into account spatial correlations. Motivated by a paper from a recent brain imaging journal. The original paper proposed an anisotropic spatial weighting function based on a flat stencil. I extended it to be both isotropic and to provide a weighted correlation based on distances.

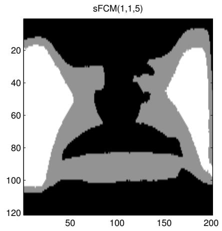
FCM and sFCM provide much better segmentation, and sFCM allows for a tunable smoothing to be included.

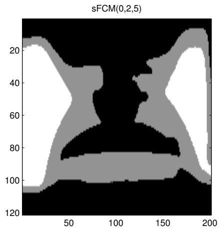
Experimental jet data

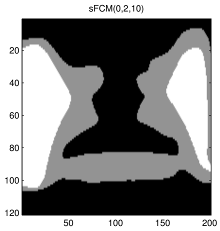




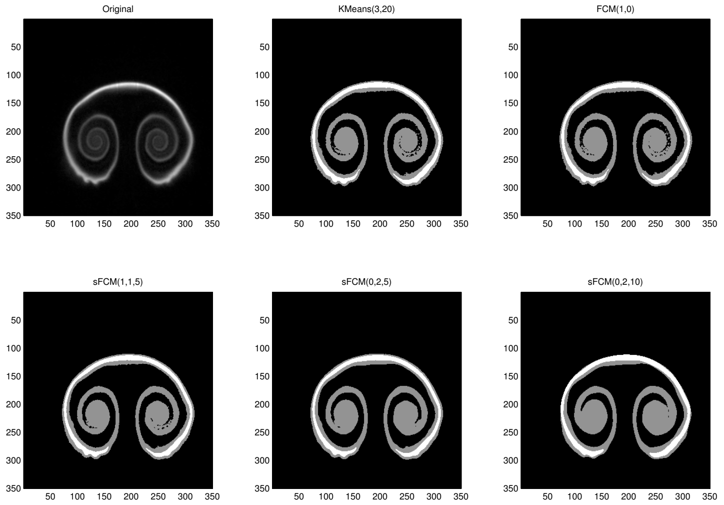




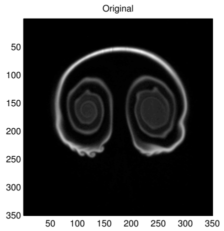


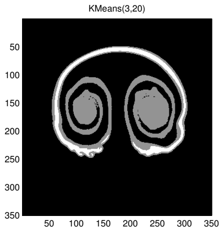


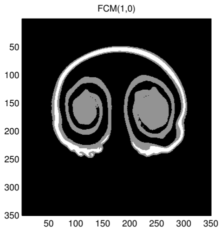
Shocktube early time

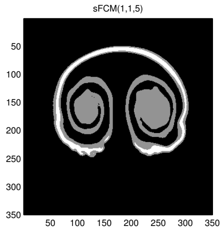


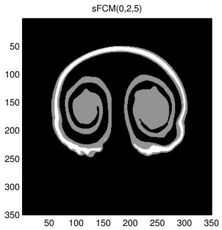
Shocktube later time

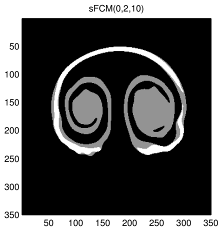




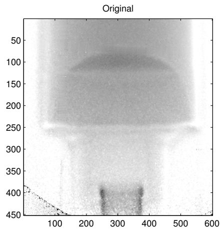


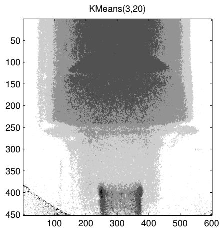


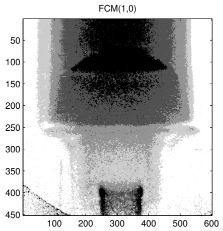


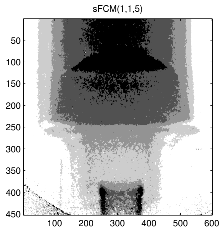


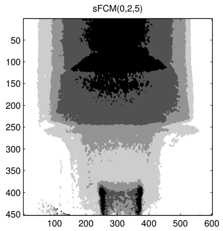
Wave collider 1

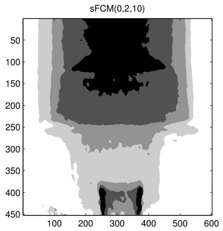




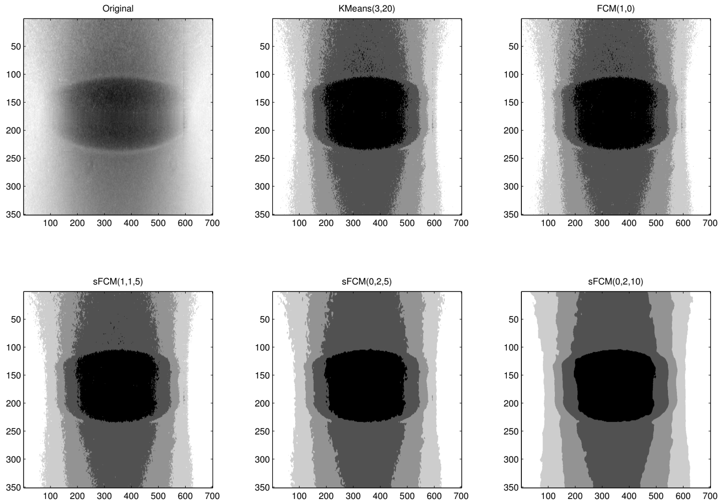




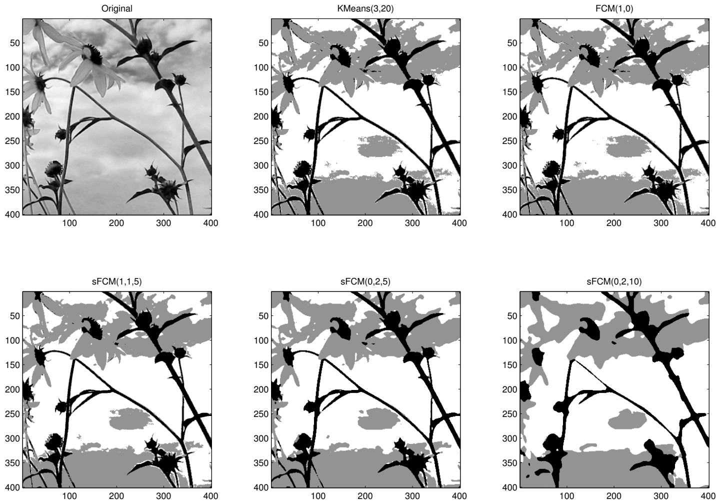




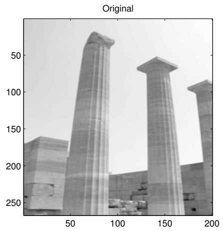
Wave collider 2

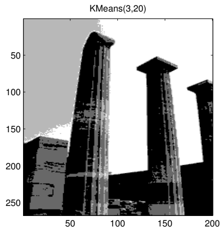


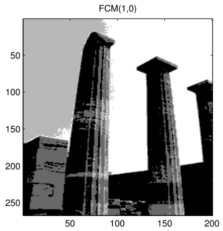
Flowers

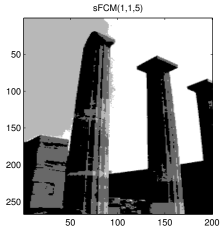

Original
KMeans(3,20)
FCM(1,0)
sFCM(1,1,5)
sFCM(0,2,5)
sFCM(0,2,10)
50
100
150
200
250
300
350
400

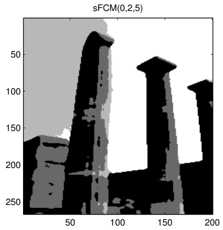
Vacation

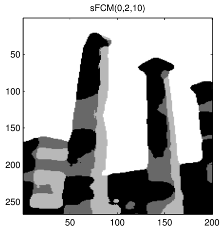

Original
50
100
150
200
250
50
100
150
200


KMeans(3,20)
50
100
150
200
250
50
100
150
200


FCM(1,0)
50
100
150
200
250
50
100
150
200

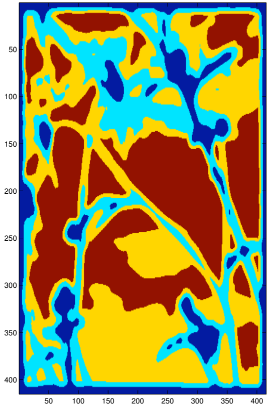

sFCM(1,1,5)
50
100
150
200
250
50
100
150
200

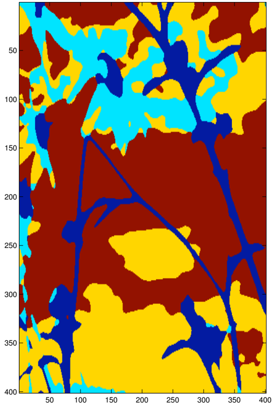

sFCM(0,2,5)
50
100
150
200
250
50
100
150
200


sFCM(0,2,10)
50
100
150
200
250
50
100
150
200

Smoothing before segmentation vs spatial segmentation

Increasing the sFCM spatial window size provides smoothing, but is **NOT** equivalent to smoothing before k-means or FCM.





Sequence similarity

Given a segmentation of the image into shape and background, we identify boundary pixels and parameterize a curve representing the estimated shape boundary. Geometric measures are computed in a sequence along the boundary. Clearly where one starts is important when comparing images.

The original DTW algorithm assumed the start point was the "correct" start point. This is not always true as the shapes evolve.

A circular DTW algorithm was implemented that has the same time and space complexity ($O(2n^2) \in O(n^2)$), but is insensitive to where the sequence starts. It finds the distance that is minimized for all possible starting positions.

Elastic sequence warping

DTW, being from speech processing, warps by removing data points to build a better fit. If we wish to warp while preserving all data, other methods must be employed. One can perform one dimensional elastic warping as an alternative.

This is partially implemented. The problem with this algorithm is the time required for the optimization loop to determine the warp coefficients.

$$dx = \sum_{j=1}^{modes} \alpha_j \sin(2\pi j)$$

Unconstrained optimization is used to find the best set of α_i's for the warp. At the current time, this method is not computationally feasible, but intelligent selection of parameters should help. (Hierarchical refinement, better optimization algorithm)

Whole image metrics

Segmentation and shape-based metrics worry me, because they require a decision to be made on where the “shape” exists within the image. Given the lack of sharp edges, there are many possible shapes depending on the parameters given to the “shape finding” algorithms.

So we would like to avoid this, and try to compute metrics that do not require shapes to be extracted.

Why?

Say I derive a shape from an image and compute a metric based on it. It is hard to say that that metric is measuring the distance between images – it's measuring the distance between structures that were *d*erived from the images. Not the images themselves.

For defensible metrics, we really should minimize the manipulations of the original data or simulations to eliminate additional sources of error and uncertainty.

Speaking of uncertainty. The deeper the processing pipeline, the more complex it is to propagate error from data/simulation to a metric. Reducing this depth is important.

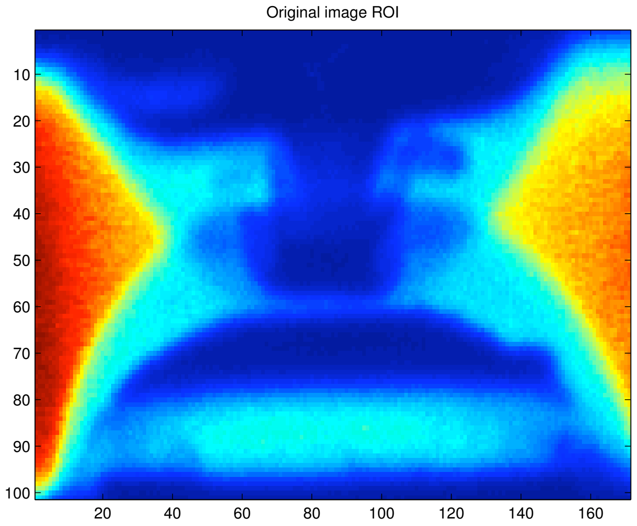
Work in progress

Currently, a few whole-image metrics are underway. At least one has results that can be shown at the current time.

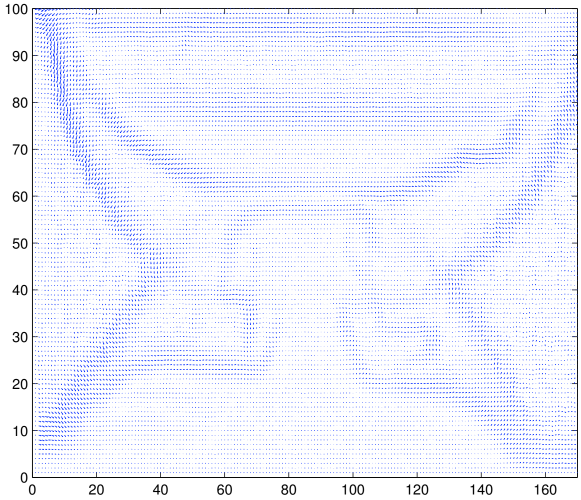
The inital step is to treat the image as a surface with heights based on pixel intensity. From this, the gradient of the image can be computed at each pixel.

This is computed with a simple square stencil that includes diagonal neighbors.

Example on experimental data



Gradient field



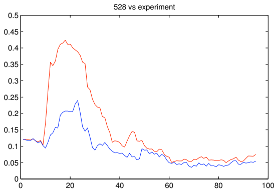
Aside: Can we use the gradient estimate as a metric?

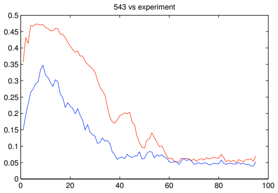
Without preprocessing the data, can we derive a metric from the gradient estimate alone? An initial experiment was performed as follows.

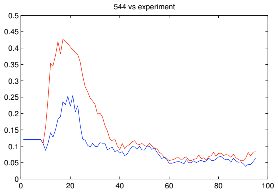
- Compute the gradient over the image.
- Create a histogram of the magnitudes of the gradient vectors.
- Normalize the histograms such that $\sum_{i=1}^{bins} bin(i) = 1.0$.
- Compare the histograms via L^2 and cDTW.

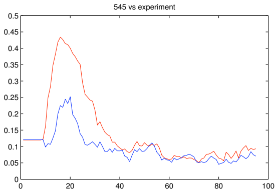
Results are inconclusive – we get numbers, but haven't gone back to the images to interpret them yet.

Some plots



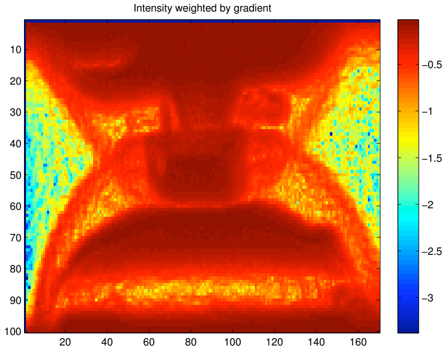






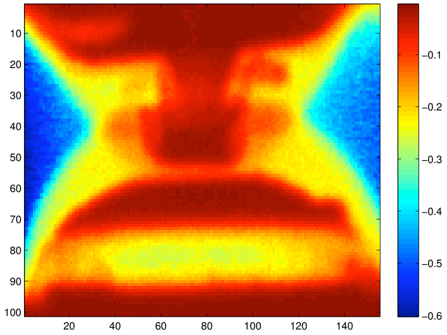
Simulations versus 6ns image are red, versus 8ns image are blue. cDTW used for histogram distance.

Intensities weighted by local gradient magnitude



There appears to be more structure evident here. Is this true, or a visual trick due to the inverted color scale?

Original image comparison, inverted color map



Nope - this is the inverted color map for the original image. By weighting pixels by the local gradient magnitude, we pull edge-like structures up away from the flat regions nearby.

Why is this interesting?

If we compute the gradient within an image that contains no clear edges, we can at least use the gradient to mark pixels as being “edge like”. In other words, a pixel with a zero gradient clearly is not at an edge, while a pixel with a moderate magnitude gradient is “near” an edge.

By using the gradients as real numbers, we don’t have to threshold and pick a value where we decide an edge “is”. That’s like segmentation, and in that case, where we pick the edge value can cause fluctuations in the subsequent metrics. This induces some undesirable sensitivities to image processing parameters that are not what we wish to measure.

Discussion

Clearly this isn't the last step, but it is interesting to consider as a preprocessing step. Instead of assigning pixels weights based on their place in the global context (ie, segmentation), we are doing so locally based on the local topology of the image.

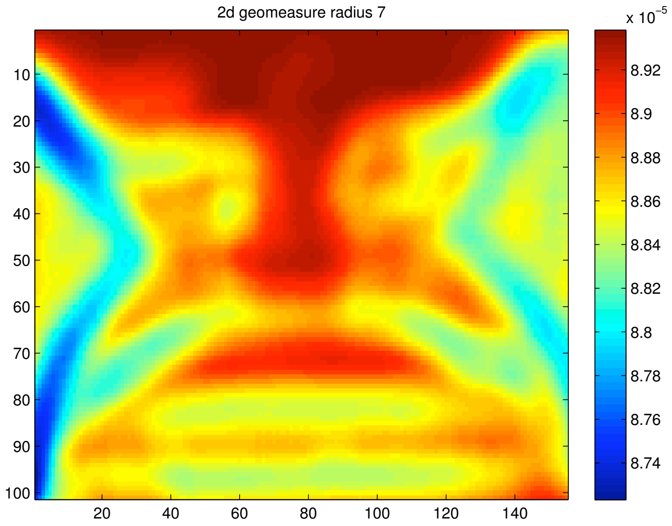
If we're looking for interfaces, shocks, or other edge-like structures, this allows us to do so. Furthermore, we can not make a binary decision (edge/not edge), but we can label pixels based on their likelihood (not in a strictly probabalistic sense) of participating in an edge-like structure.

Moving towards whole image geometric measures

One can also treat the image as a surface, and estimate it's properties by triangulation. Triangulation of the surface can estimate quantities such as surface area and volume.

Consider this. We can easily compute the surface area of the image via triangulation. At any pixel, we can consider a sphere of radius r. One can compute the ratio of the area of the disk of radius r to the surface area of the image that the sphere intersects. Similarly, one can treat the image as a height field, and compute the ratio of the volume of a sphere centered at a pixel and mean pixel value as height, versus the volume of the intersected height field.

First cut whole image geomeasure



Discussion of 2d geomeasure

What was that picture? At each pixel, we plot:

$$\frac{\text{Pixels in disc of radius r}}{\text{Surface area of image covered by disc of radius r}}$$

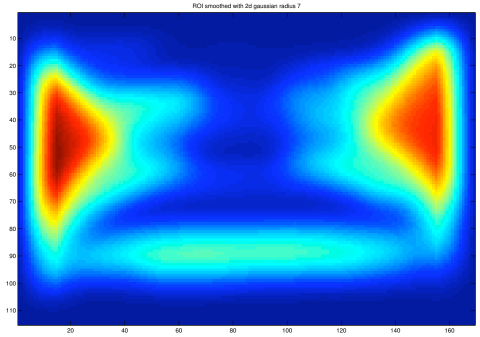
What does this tell us?

- Flat regions of image, this ratio is close to one.
- Bumpy regions, this ratio is less than one.
- Edges with high gradient region much less than one.
- Wavy edges have ratio less than straight edges.

I believe this is the 3d "equivalent" (in spirit at least) of the shape measure of circle circumference to contained arc length. In 3d we consider spheres/ellipsoids, so instead of a 1d boundary, we get 2d surfaces and discs.

But wait...

That looks like a smoothing filter output, right? Wrong.



This is approximately the same region of the image after applying a Gaussian smoothing with a radius of 7 pixels, the same as the radius used for the geomeasure computation.

Comparing whole image geomeasures

How do we compare these whole image geomeasures? Not sure yet. Warping is an option. So is basic L2, although that will likely suffer from registration mismatches. L2 discussed in IS06 slides for shocktube data on multiscale 1D geomeasures.

It's really an open question as of now as to how to best compare these full image geomeasures.

Warping revisited

Tom and I spent some time at the SIAM Imaging Science conference this May in Minneapolis talking to one of the experts in the field of image warping and registration (Jan Modersitzki).
At one time, Tom and I worked on a DDMA project entitled "TSWarp", in which we explored the possibility of using image warping maps to compute warp energies that can be used as distance metrics between images.

- Curvature warping
- Fluid warping
- Elastic warping (the one I worked most on)

Warping (2)

Issues arise.

- We wish to avoid circular problems. We don't want to do fluid registration that relies on evolving a boundary that amounts to essentially what is being simulated.
- Conservation properties. Warping requires interpolation of an image under a warp mapping in order to do a comparison of the warped image to that which is being compared. Linear or other interpolation methods do not obey conservation laws. Conservative remappers are computationally nasty.
- The data sets we are considering require warping operators to allow shear flows and twisting that makes optimization loops both computationally difficuly, and prone to getting stuck in local minima.

Warping (3)

New methods based on particle warping that boil down to fluid warping with some terms eliminated may have some promise.

At IS06, a student from U. Waterloo presented a paper on this. Currently awaiting a draft or some code to try it out.

In the meantime, we are resurrecting our TSWarp algorithms that were investigated under Jim Kamm's LDRD and applied to the shocktube and wavecollider data sets.

- Why did they get shelved? Not appropriate for shocktube due to flow characteristics. Furthermore, elastic warping on whole image has a REALLY nasty optimization step.
- Mass conserving interpolator desired also. This also proved to be a computationally painful algorithm.

Nuance formation

DDMA also formed a new working group that focuses on detecting data nuances. Much of our work has focused on shape metrics, denoising, warping, etc... The jets data is unique in that what we are looking for is buried in the data - it's not a large scale obvious difference.

Shape metrics do work – registration problems have shown that, as have other data sets. Furthermore, a first glance at the shape data that comes out of the four new simulations looks promising for shape comparisons.

Why are we looking outside the shape realm? We're avoiding the "have hammer, world is full of nails" syndrome. Sometimes you need a hacksaw if you really can't turn the problem into a nail.